Cricket Ball Detection & Tracking System

Detailed Technical Report

1. Introduction

This project aims to build a **robust cricket ball detection and tracking system** using video input from a **single static camera**. The system detects the ball in each frame, computes its centroid, tracks it across time, handles missed detections, and produces both a **per-frame annotation file** and a **visualized trajectory video**.

The design emphasizes:

- Reproducibility
- Clear failure handling
- Industry-standard modelling choices
- Practical robustness over theoretical complexity

---

2. Modelling Decisions

2.1 Detection Model Selection

Model chosen: YOLOv8 (single-class detection)

Reasoning:

- Cricket ball is a **small, fast-moving object**
- YOLOv8 provides:
    - Strong small-object detection
    - Real-time inference
    - Easy training and deployment
- Widely accepted in production CV pipelines

Output used from detector:

- Bounding box (x1, y1, x2, y2)
- Centroid computed as:

$$(x_c, y_c) = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right)$$

2.2 Tracking & Temporal Smoothing

Tracker used: Kalman Filter (constant velocity model)

Why Kalman Filter?

- YOLO detections are frame-independent and noisy
- Kalman filter:
    - Smooths jitter
    - Predicts motion during occlusion
    - Maintains trajectory continuity

State vector:

$$[x, y, v_x, v_y]$$

This allows the tracker to estimate both position and motion of the ball.

2.3 Why Not Deep Tracking Models?

- The problem constraints assume:
    - Single object
    - Static camera
- Kalman filter is:
    - Simpler
    - More interpretable
    - Easier to debug
- Deep trackers (DeepSORT, ByteTrack) were unnecessary overhead

---

3. Fallback Logic & Robustness

The system is explicitly designed to **not hallucinate detections**.

3.1 Detection Fallback

Scenario	System Behavior
Ball detected	Use YOLO centroid → Kalman update
Ball not detected	No update, visibility = 0
Short occlusion	Kalman predicts internally
Long occlusion	Trajectory naturally stops

3.2 Visibility Flag Logic

Each frame outputs:

- visible = 1 → Ball confidently detected
- visible = 0 → Ball not visible / occluded

Coordinates for invisible frames:

x = -1

y = -1

This avoids introducing false ground truth.

---

4. Assumptions Made

The following assumptions were explicitly made and documented:

1. **Single ball present** at any time
2. **Static camera** (no pan/tilt/zoom)
3. **Reasonable lighting conditions**
4. Ball visible for at least short continuous segments
5. No extreme motion blur

These assumptions match real-world broadcast cricket analytics scenarios.

---

5. Example Outputs

5.1 Annotation File (CSV)

frame,x,y,visible

0,512.3,298.1,1

1,518.7,305.4,1

2,-1,-1,0

3,530.1,315.2,1

5.2 Processed Video

- 🔴 Red dot → current centroid
- 🔵 Blue line → trajectory history

The trajectory is cumulative and visually smooth due to Kalman filtering.

---

6. Issues Encountered & Fixes Applied

This section documents **real issues** and **how they were solved**.

Issue 1: False Positives (Shoes, Gloves, White Objects)

Cause:

- Generic pretrained YOLO model
- Visually similar objects to cricket ball

Fixes Applied:

- Trained YOLO on **domain-specific cricket ball dataset**
- Increased confidence threshold (CONF_THRESH)
- Reduced IOU tolerance

Result:

- Significant reduction in false detections
- Cleaner trajectory

Issue 2: Trajectory Jitter

Cause:

- Frame-wise detection noise
- Small bounding box variations

Fix:

- Introduced Kalman filter smoothing
- Tuned measurement noise (R) and process noise (Q)

Result:

- Smooth, physically plausible trajectory
- Stable centroid path

Issue 3: Missed Detections During Occlusion

Cause:

- Ball temporarily hidden behind player or pitch

Fix:

- Kalman prediction maintained internal state
- Visibility flag set to 0 (no fake output)

Result:

- No false annotations
- Trajectory continuity preserved visually

Issue 4: .mov File Incompatibility on Windows

Cause:

- OpenCV codec limitations on Windows

Fix:

- Standardized input to .mp4 using FFmpeg
- Output always generated as .mp4

Result:

- Fully reproducible pipeline
- No silent failures

Issue 5: Output File Naming Errors

Cause:

- Dynamic filename accidentally included directory separators

Fix:

- Used os.path.basename() and os.path.splitext()
- Sanitized output filenames

Result:

- Correct CSV and video output paths
- No filesystem errors

---

7. Model Performance Improvements

Stage	Improvement
Initial run	Pretrained YOLO, many false positives
After training	Domain-specific ball detection
After tuning	Higher precision
After tracking	Smooth trajectory
Final system	Stable + reproducible

---

8. Hyperparameter Calibration

Parameter	Value	Reason
Confidence Threshold	0.3–0.4	Reduce false positives
Kalman R	Increased	Reduce jitter
Kalman Q	Low	Smooth motion assumption

---

9. Reproducibility

The system is fully reproducible using:

pip install -r requirements.txt

python code/main.py

Outputs:

- annotations/output_<video>.csv
- results/trajectory_<video>.mp4

---

10. Limitations & Future Work

- Multi-ball handling
- Bounce detection
- Spin estimation
- 3D trajectory reconstruction
- Multi-camera fusion

---

11. Conclusion

This project demonstrates a **practical, production-aligned approach** to cricket ball tracking. By combining deep learning detection with classical filtering, the system achieves robustness, interpretability, and reproducibility — key requirements for real-world sports analytics systems.